Pet clinic mandatory assignment report

Krisztian Szabo

October 30, 2019

Contents

1 Introduction

2 Implementing the changes

I already had my development environment set up, with JDK, Maven, and git correctly installed and added to the environment variables. Cloning the repository to a local folder was also trivial.

The first real step in tackling this assignment was to identify where to make changes within the codebase. By default, the Pet Clinic web application uses **hsqldb** as data storage. I found out that it works as an in-memory database, which creates its structure and populates it with data at runtime. So the place to modify the table structure is in the **src/main/resources/db/hsqldb/schema.sql** file and adding data to populate it in the **src/main/resources/db/hsqldb/data.sql** file.

The next thing to be modified was to add a new menu entry on the navigation bar for the future view that needed to be created. I found the navigation bar as a thymeleaf fragment in the **src/main/resources/templates/fragments/layout.html** file. I added an additional entry and pointed it to "/drugs" mapping, which for the moment does not yet exist.

Next task was to define the **Drug** class, which in our case should extend the existing **NamedEntity** class. It has to define the **type**, **batchNumber**, and **expiryDate** attributes, and bind them to their respective SQL columns. The **id** and **name** attributes are taken care of in the **BaseEntity** and **NamedEntity** superclasses, respectively.

Then, I defined the **DrugRepository** interface, which defines some basic methods for retrieving and saving **Drug** entities from and to the database. Having done this, I moved to writing the **DrugController** class. In it, I added an instance of **DrugRepository** and defined the GET mapping for "/drugs". All this does is it inserts a list of all the available **Drug** entities into the model, and returns the **src/main/resources/templates/drugs/drugs.html** view. Next I went ahead and actually created the view, which consists of a simple html table which is populated by thymeleaf by iterating over the list of **Drug** entities that was sent from the controller through the model.

After getting this to work, I realized that it would make logical sense to define a many-to-many relationship between the newly created drugs and the animal types, as one drug can most likely be used for several animals. To achieve this, I added a new table to the **schema.sql** file, that only serves

the purpose of mapping this relationship, by the means of having only two columns, which are each a foreign key pointing to the **id** columns of the **drugs** and **types** tables, respectively. I then went ahead and modified the **Drugs** POJO to hold, instead of a single **PetType** attribute, a **List¡PetType¿**. The last change needed to be made was in the view, for each drug displayed in the table, to iterate over all of its associated **PetType**s and show them in a nicely formatted string.